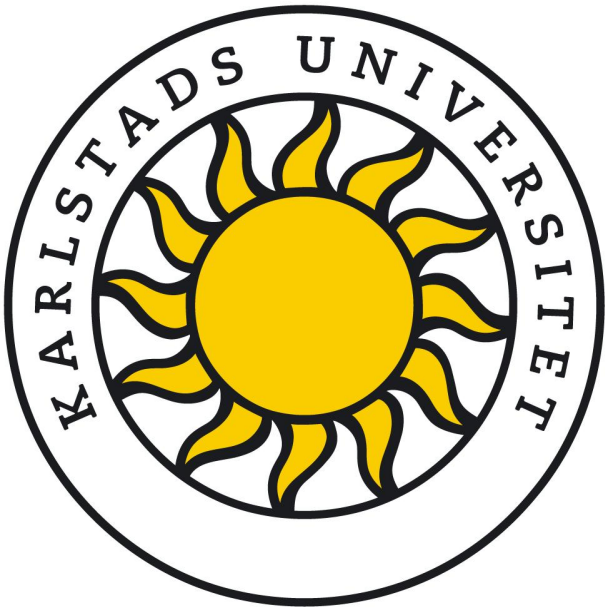
Laborationsrapport

Funktionsanalys



Johan Brekstad Wijk

1996-05090753

Karlstad Universitet

18 November 2022

1 Introduktion

I denna laboration undersöks olika sök- och sorteringsalgoritmer med syfte att analysera hur effektiva de är samt hur mycket körtiden ökar beroende på mängden tal som ska sorteras. Resultatet demonstreras via ett konsol-baserat program där användaren via en meny kan få fram data kring hur de olika algoritmerna presterar.

2 Algoritmer

Programmet är en vidareutveckling på startkoden som tilldelats och utvecklingen har skett löpande under ca två veckors tid.
Algoritmerna som undersöktes var följande:

Bubble sort

Bubble sort är en väldigt enkel algoritm som i en nästlad for-loop kollar första talet med det andra närliggande talet och byter plats om det första är större än det andra och fortsätter vidare så tills arrayen är slut, därefter börjar den om och kör så tills hela arrayen är sorterad. I en optimerad variant av algoritmen så läggs en boolean till som blir true om det sker ett byte. Sker det inget byte betyder det att arrayen är sorterad och funktionen avbryter.
Best case för bubble sort är när listan är sorterad i storleksordning, worst case är listan sorterad störst till minst och i average case så är listan slumpmässigt sorterad.
Tidskomplexitet:
Best case: O/n
Worst case: O/n^2
Average case: O/n^2

Insertion sort

Insertion sort fungerar likt bubble sort där två närliggande tal jämförs och byter plats om det första är större än det andra talet. Om det sker ett byte så kommer traverseringen att byta håll och byta plats på talen tills det ligger sorterat. Därefter fortsätter sorteringen tills hela listan är sorterad. Best case för insertion sort är när listan är sorterad i storleksordning, worst case är när listan är sorterad störst till minst och i average case är listan slumpmässigt sorterad.
Tidskomplexitet:
Best case: O/n
Worst case: O/n^2
Average case: O/n^2

Quicksort

Quicksort är en rekursiv “divide and conquer”-algoritm vilket betyder att den delar in data i mindre delar och kallar sedan rekursivt på sin egna funktion. Det första som händer i algoritmen är att ett så kallat “pivot”-element väljs ut. Sedan jämförs resterande värden mot pivot-värdet och sorteras i två olika högar, en större och en lägre. När det är klart så är pivot elementet sorterat och klart och funktionen kallar på sig själv igen fast nu med nya värden som parametrar. På så sätt delas listan ned i mindre delar och tillslut är listan sorterad. Best case för quicksort algoritmen är då de två olika partitionerna som pivot talet bildar är lika stora. Worst case uppstår när partitionerna blir så obalanserade som möjligt och average case testas på en slumpmässig pivot.
Tidskomplexitet:
Best case:O/nlogn
Worst case: O/n^2
Average case: O/nlogn

Linear search

Linear search är en väldigt simpel sökalgoritm som tar emot en pekare till en lista, storleken på listan och ett tal som ska sökas efter. Väl inne i funktionen så startar en for-loop som går igenom hela listan och med hjälp av en if-sats, kollar om talet är lika med talet på positionen i arrayen, om så returnerar funktionen true, om inte talet hittas och hela listan har traverseras så returnerar funktionen false. Best case är om talet som söks är det första talet i listan. Worst case är när listan är sorterad i storleksordning och det är det sista talet som söks.
Tidskomplexitet:
Best case: O/1
Worst case: O/n
Average case: O/n

Binary search

Binary search går till som så att listan delas på mitten för att se om talet finns där. Ligger talet där så returnerar funktionen true då talet är hittat. Om talet inte finns i mitten av listan så jämförs talet för att se om det är större eller mindre än det mittersta talet. Är talet man söker efter mindre så kommer programmet att göra samma sak igen fast endast på den en halvan som innehåller de mindre talen. Är talet större så hade samma sak skett fast på den sida med de större talen. Detta pågår tills talet är hittat. Best case för binary search är om
Tidskomplexitet:
Best case: O/1.
Worst case:O/logn
Average case: O/logn

3 Analys

Av de tre sorteringsalgoritmen så är Quicksort snabbare än både Bubblesort och Insertion sort. Bubble och Insertion sort har en under average case en tidskomplexitet på O(n^2) vilket innebär att för varje element i listan så tar det lika många gånger talet sig själv i operationer att utföra. Quicksort har, under average case, en tidskomplexitet på O(nlogn) och i worst case en tidskomplexitet på O(n^2) vilket innebär att Quicksort är det snabbare och bättre alternativet. (se Bilagor för resultat)
Vad som även kunde analyseras var att både linear search och binary search var extremt snabba, de var så snabba att de var tvunget att implementera extra test omgångar för att överhuvudtaget få ut någon användbar data. Linear search är extremt snabb speciellt vid best case samt vid mindre listor. Vid worst case så lämpar sig dock binary search.

4 Problem

Ett stort problem som uppmärksammades var att datan från de två sök algoritmerna inte var användbar (se bilaga Problem). Problemet här var att algoritmerna är alldeles för snabba och programmet inte hinner att ta tid. Problemet löstes på så sätt att istället för att kalla på funktionerna endast fyra gånger så implementerades en for-loop som itererade 100 000 gånger. Detta resulterade i mer användbar data.

5 Sammanfattning

Sammanfattningsvis så har laborationen varit lärorik och förståelse kring de algoritmer som analyserats har stärkts. Laborationen har även givit insikt till att varje algoritm har sin plats och att det inte finns en algoritm som passar bäst till allt.

6 Bilaga

Problem

```
*********************************************************************************
                              Linear search:best
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Size    Time T(s)       T/logn          T/1             T/n
512     0.0000000000    0.000000e+00    0.000000e+00    0.000000e+00
1024    0.0000000000    3.606738e-13    2.500000e-12    2.441406e-15
2048    0.0000000000    0.000000e+00    0.000000e+00    0.000000e+00
4096    0.0000000000    0.000000e+00    0.000000e+00    0.000000e+00
8192    0.0000000000    2.774414e-13    2.500000e-12    3.051758e-16
16384   0.0000000000    0.000000e+00    0.000000e+00    0.000000e+00
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
input> o
*********************************************************************************
                              Binary search:best
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Size    Time T(s)       T/logn          T/1             T/n
512     0.0000000000    0.000000e+00    0.000000e+00    0.000000e+00
1024    0.0000000000    3.606738e-13    2.500000e-12    2.441406e-15
2048    0.0000000000    0.000000e+00    0.000000e+00    0.000000e+00
4096    0.0000000000    6.011229e-13    5.000000e-12    1.220703e-15
8192    0.0000000000    0.000000e+00    0.000000e+00    0.000000e+00
16384   0.0000000000    0.000000e+00    0.000000e+00    0.000000e+00
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Bubble sort

```
*************************************************************************
                          bubble sort:best
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Size    Time T(s)       T/logn          T/n             T/nlogn
512     0.0000017500    2.805240e-07    3.417969e-09    5.478985e-10
1024    0.0000030000    4.328085e-07    2.929688e-09    4.226646e-10
2048    0.0000057500    7.541360e-07    2.807617e-09    3.682305e-10
4096    0.0000105000    1.262358e-06    2.563477e-09    3.081929e-10
8192    0.0000245000    2.718925e-06    2.990723e-09    3.319001e-10
16384   0.0000530000    5.461631e-06    3.234863e-09    3.333515e-10
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
*************************************************************************
                          bubble sort:worst
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Size    Time T(s)       T/nlogn         T/n^2           T/n^3
512     0.0011087500    3.471328e-07    4.229546e-09    8.260831e-12
1024    0.0037092500    5.225895e-07    3.537416e-09    3.454508e-12
2048    0.0135455000    8.674550e-07    3.229499e-09    1.576904e-12
4096    0.0532732500    1.563661e-06    3.175333e-09    7.752278e-13
8192    0.2139977500    2.899015e-06    3.188815e-09    3.892596e-13
16384   0.8602925000    5.410940e-06    3.204839e-09    1.956079e-13
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
*************************************************************************
                          bubble sort:average
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Size    Time T(s)       T/nlogn         T/n^2           T/n^3
512     0.0008070000    2.526595e-07    3.078461e-09    6.012619e-12
1024    0.0031620000    4.454884e-07    3.015518e-09    2.944842e-12
2048    0.0115637500    7.405435e-07    2.757013e-09    1.346198e-12
4096    0.0517755000    1.519699e-06    3.086060e-09    7.534327e-13
8192    0.2303780000    3.120917e-06    3.432900e-09    4.190551e-13
16384   0.9695727500    6.098274e-06    3.611940e-09    2.204553e-13
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Insertion sort

```
********************************************************************************
                              Insertion sort:best
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Size    Time T(s)       T/logn          T/n             T/nlogn
512     0.0000030000    4.808983e-07    5.859375e-09    9.392546e-10
1024    0.0000070000    1.009887e-06    6.835937e-09    9.862173e-10
2048    0.0000130000    1.705003e-06    6.347656e-09    8.325211e-10
4096    0.0000227500    2.735109e-06    5.554199e-09    6.677513e-10
8192    0.0000622500    6.908290e-06    7.598877e-09    8.432971e-10
16384   0.0001112500    1.146427e-05    6.790161e-09    6.997237e-10
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
********************************************************************************
                              Insertion sort:worst
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Size    Time T(s)       T/nlogn         T/n^2           T/n^3
512     0.0006277500    1.965390e-07    2.394676e-09    4.677102e-12
1024    0.0016007500    2.255268e-07    1.526594e-09    1.490815e-12
2048    0.0060337500    3.864019e-07    1.438558e-09    7.024209e-13
4096    0.0235565000    6.914235e-07    1.404077e-09    3.427922e-13
8192    0.0927985000    1.257136e-06    1.382805e-09    1.687995e-13
16384   0.3738832500    2.351595e-06    1.392824e-09    8.501121e-14
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
********************************************************************************
                              Insertion sort:average
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Size    Time T(s)       T/nlogn         T/n^2           T/n^3
512     0.0002590000    8.108898e-08    9.880066e-10    1.929700e-12
1024    0.0008772500    1.235942e-07    8.366108e-10    8.170027e-13
2048    0.0030142500    1.930328e-07    7.186532e-10    3.509049e-13
4096    0.0118232500    3.470326e-07    7.047206e-10    1.720509e-13
8192    0.0464380000    6.290929e-07    6.919801e-10    8.447023e-14
16384   0.1842382500    1.158794e-06    6.863410e-10    4.189093e-14
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Quicksort

```
**************************************************************************
                            Quicksort:best
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Size    Time T(s)       T/n             T/nlogn         T/n^2
512     0.0000552500    1.079102e-07    1.729794e-08    2.107620e-10
1024    0.0001005000    9.814453e-08    1.415926e-08    9.584427e-11
2048    0.0002280000    1.113281e-07    1.460114e-08    5.435944e-11
4096    0.0006132500    1.497192e-07    1.799993e-08    3.655255e-11
8192    0.0016612500    2.027893e-07    2.250486e-08    2.475455e-11
16384   0.0051972500    3.172150e-07    3.268889e-08    1.936127e-11
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
**************************************************************************
                            Quicksort:worst
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Size    Time T(s)       T/nlogn         T/n^2           T/n^3
512     0.0005370000    1.681266e-07    2.048492e-09    4.000962e-12
1024    0.0018595000    2.619816e-07    1.773357e-09    1.731794e-12
2048    0.0065127500    4.170771e-07    1.552761e-09    7.581839e-13
4096    0.0252297500    7.405362e-07    1.503810e-09    3.671412e-13
8192    0.0996252500    1.349617e-06    1.484532e-09    1.812173e-13
16384   0.3986157500    2.507154e-06    1.484959e-09    9.063473e-14
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
**************************************************************************
                            Quicksort:average
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Size    Time T(s)       T/n             T/nlogn         T/n^2
512     0.0000652500    1.274414e-07    2.042879e-08    2.489090e-10
1024    0.0001037500    1.013184e-07    1.461715e-08    9.894371e-11
2048    0.0002557500    1.248779e-07    1.637825e-08    6.097555e-11
4096    0.0006560000    1.601563e-07    1.925472e-08    3.910065e-11
8192    0.0016612500    2.027893e-07    2.250486e-08    2.475455e-11
16384   0.0050587500    3.087616e-07    3.181777e-08    1.884531e-11
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Tidskomplexitet